1

Defining Classes

Consider the following scenario:

A person's kennitala is assigned at birth, and remains fixed afterwards. A person's name is assigned later, and can be changed anytime. Anyone may know a person's name and kennitala.

Implement a class *Person* with attributes, appropriate get/set methods and an appropriate constructor (with appropriate visiblities) in Java to reflect this scenario. All attributes should have a *String* data type.

Write your answer in the box below. Changes are saved automatically.

Test case #	Input	Expected output
-		

2 Using Objects

Assume you are given a public class *Card* that represents a poker card, as well as a public class *Pile* that represents a pile of cards in a game, and provides the following public methods:

- *Pile()* constructs a new pile filled with random cards
- *Card draw()* draws one card off the pile and returns it
- *void discard(Card c)* places the given card onto the pile
- *boolean isEmpty()* returns true if the pile is empty, or false otherwise

Assuming that the implementations of the above classes already exist, implement a public void method *test()* in Java that uses these classes in order to:

- construct two piles *p* and *q*
- until pile *p* is empty, draw cards from pile *p* and discard them onto pile *q*

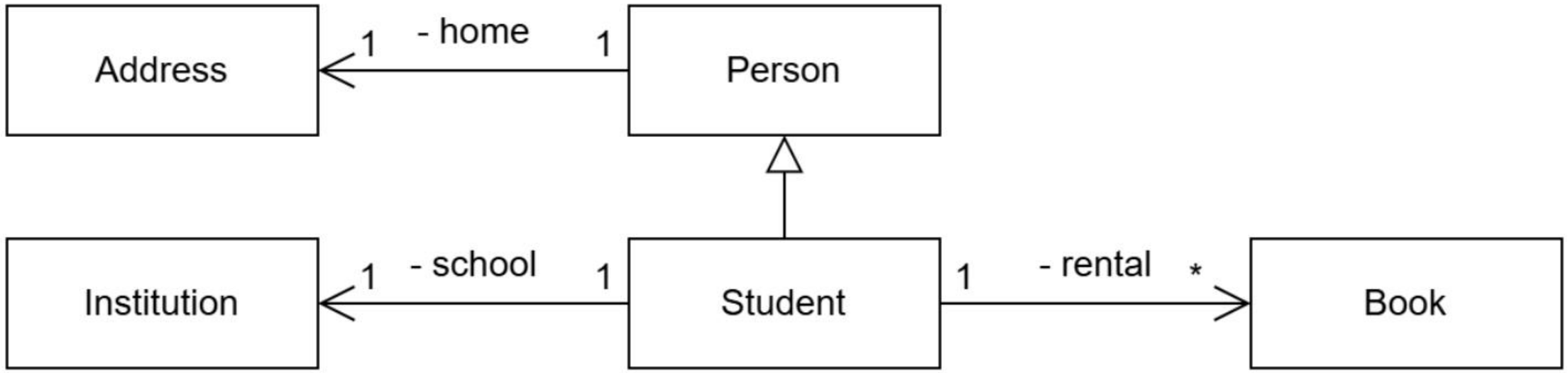
Write your answer in the box below. Changes are saved automatically.

Test case #	Input	Expected output
-		

3

Class Relations

Consider the following UML class diagram:



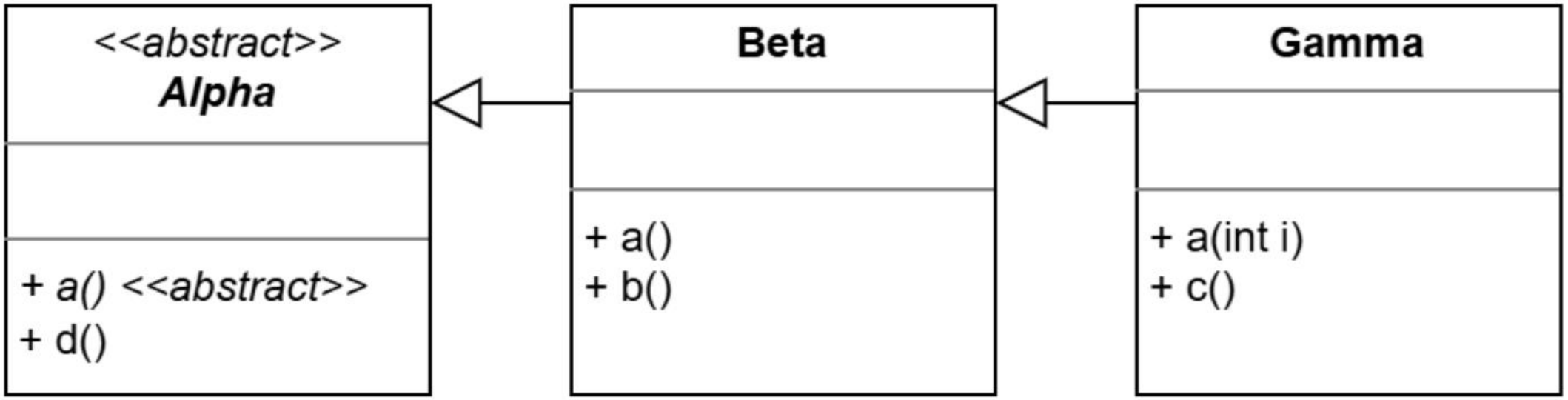
Implement the *Student* class (including its attributes, but no methods) in Java in accordance with this class diagram.

Write your answer in the box below. Changes are saved automatically.

Test case #	Input	Expected output
-		

4 Inheritance

Consider the following UML class diagram:



Assuming these classes are implemented in Java as shown above, indicate which implementation of *a()* will be invoked by the statement ***obj.a()***, or if a compiler error will occur, if *obj* is instantiated as follows:

	a() in Alpha	a() in Beta	a() in Gamma	Compiler error
Alpha obj = new Alpha();	○	○	○	◉
Beta obj = new Beta();	○	◉	○	○
Alpha obj = new Beta();	○	◉	○	○
Gamma obj = new Beta();	○	○	○	◉
Alpha obj = new Gamma();	○	○	◉	○

(1 point per correct answer; max. 5 points)

5

Implementing Interfaces

The Java Class Library's interface *Comparable<T>* declares a method *public int compareTo(T obj)* that returns

- a negative value if this object is smaller than the given *obj*
- a positive value if this object is bigger than the given *obj*
- zero otherwise

where *T* is the type (class) of the objects to be compared.

Modify the following Java class representing a person so that it implements the *Comparable* interface such that people are ordered

- by birth year
- or if that's the same, then by birth month
- or if that's also the same, then by birth day.

Write your answer in the box below. Changes are saved automatically.

Test case #	Input	Expected output
-		

6 Polymorphism

The Java Class Library's interface *Comparable<T>* declares a method *public int compareTo(T obj)* that returns

- a negative value if this object is smaller than the given *obj*
- a positive value if this object is bigger than the given *obj*
- zero otherwise

where *T* is the type (class) of the objects to be compared.

Implement a method *countBiggerThan* that expects the following parameters:

- an array of *Comparable* objects to examine
- a single *Comparable* object serving as a baseline

The method shall return an integer value that indicates how many objects in the array are bigger than the baseline object.

Write your answer in the box below. Changes are saved automatically.

Test case #	Input	Expected output
-		